**LAB EXERCISE 1**

**BELLMAN FORD ALGORITHM IMPLEMENTATION**

**SOURCE CODE :**

**#include** <iostream>

**#include** <fstream>

**#include** <string>

**#include** <sstream>

**#include** <stdlib.h>

**#include** <stdio.h>

**#include** <limits.h>

//edge representation of the graph

**struct** Edge

{

**int** src,dest,weight;

};

//graph representation

**struct** Graph

{

**int** V,E;

**struct** Edge \*edge ;

};

//create a graph

**struct** Graph\* **creategraph**(**int** V,**int** E)

{

**struct** Graph\* newgraph=**new** Graph();

newgraph->E=E;

newgraph->V=V;

newgraph->edge=**new** Edge[E];

**return** newgraph;

}

//declare a global variable i

**int** i;

//add an edge to graph

**struct** Graph\* **addedge**(Graph \*graph,**int** src,**int** dest,**int** weight)

{

**if**(i==graph->E)

**return** 0;

**else**

graph->edge[i].src=src;

graph->edge[i].dest=dest;

graph->edge[i].weight=weight;

i++;

**return** graph;

}

//print distance array

**void** **displaydist**(**int** arr[], **int** size) {

**int** i;

std::ofstream outfile;

outfile.open("result.txt",std::ios\_base::*app*);

**for**(i = 0; i < size; i ++) {

outfile<<arr[i]<<",";

}

outfile.close();

}

//print path

**void** **display**(**int** arr[], **int** size) {

**int** i;

std::ofstream outfile;

outfile.open("result.txt",std::ios\_base::*app*);

**for**(i = 0; i < 1; i ++)

{

outfile<<arr[i];

}

//if(arr[i]!=0)

**for**(i=1;i<size;i++)

{ **if**(arr[i]!=0)

{

outfile<<"->"<<arr[i];

}

}

outfile<<"\n";

outfile.close();

}

//bellman ford implementation

// The main function that finds shortest distances from src to

// all other vertices using Bellman-Ford algorithm. The function

// also detects negative weight cycle

**void** **BellmanFord**(**struct** Graph\* graph, **int** src)

{

**int** V = graph->V;

**int** E = graph->E;

**int** dist[V];

**int** pre[V];

// Step 1: Initialize distances from src to all other vertices

// as INFINITE

**for** (**int** i = 0; i < V; i++)

{

dist[i]= INT\_MAX;

pre[i]=0;

}

dist[src] = 0;

// Step 2: Relax all edges |V| - 1 times.

**for** (**int** i = 1; i < V-1;i++)

{

**for** (**int** j = 0; j <E;j++)

{

**int** u = graph->edge[j].src;

**int** v = graph->edge[j].dest;

**int** weight = graph->edge[j].weight;

**if** (dist[u] != INT\_MAX && dist[v]>dist[u] + weight )

{

dist[v] = dist[u] + weight;

if(dist[v]==dist[u])

cout<<”Iteration”<< pre[v]=u;

display(pre,V);

}

}

}

// Step 3: check for negative-weight cycles.

**for** (**int** j = 0; j < E; j++)

{

**int** u = graph->edge[j].src;

**int** v = graph->edge[j].dest;

**int** weight = graph->edge[j].weight;

**if** (dist[u] != INT\_MAX && dist[v]>dist[u] + weight)

**printf**("Negative Loop detected");

}

displaydist(dist,V);

**return**;

}

//use command line arguments to provide the file name of the graph as an input parameter to the complied program

**int** **main**(**int** argc, **char** \*argv[]) {

**int** array[500][500];

std::ifstream file( argv[1] );

std::string line;

**int** dest = 0;

**int** src = 0;

**struct** Graph\* graph;

**while**(std::**getline**(file,line,'\n'))

{

std::istringstream iss( line );

std::string weight;

**while**( std::**getline**( iss, weight, ',' ) )

{

array[src][dest] = **atoi**( weight.c\_str() );

// std::cout << weight<<" ";

dest = dest+1;

}

src = src+1;

dest = 0;

std::cout<<std::**endl**;

}

graph=creategraph(src,src\*src);

**for**(**int** j=0;j<src;j++)

{

**for**(**int** k=0;k<src;k++)

{

**if**(array[j][k]!=0)

addedge(graph,j,k,array[j][k]);

}

}

BellmanFord(graph,0);

**return** 0;

}

**OUTPUT**

**1.N7.csv**

**Result.txt :**

0,33,13,10,24,26,38

0

0->3->1

0->3->2

0->3

0->3->4

0->3->4->5

0->3->1->6

Iteration:4

**2.N10.csv**

**Result.txt**

0,5,3,10,2,8,7,1,7,8

0

0->4->1

0->2

0->2->6->3

0->4

0->4->1->5

0->2->6

0->7

0->4->1->8

0->2->6->9

Iteration:4

**3.N20.csv**

**Result.txt**

0,4,6,7,6,4,10,7,7,4,9,7,8,6,5,3,5,7,4,5

0

0->15->1

0->15->1->2

0->5->19->3

0->4

0->5

0->18->6

0->5->7

0->15->9->8

0->15->9

0->15->9->10

0->18->11

0->15->12

0->15->13

0->5->14

0->15

0->18->16

0->15->9->17

0->18

0->5->19

Iteration:4

**4.neg.csv**

**Result.txt**

0

0->1

2->1

Negative loop detected

**PROGRAM CODE EXPLANATION**

1.As input file name should be an input parameter to the complied program “./Ford N7.csv”,command line arguments have been used to get input.

2.Structures were defined to represent edge, graph, for creating a graph and to add an edge to the graph

3.A separate function for Bellman ford algorithm was created. In which, the distance from all vertices to the source is defined as infinity and for source it is provided as zero.

4.For each edge the following is executed:

If dist[v] > dist[u] + weight of edge uv, then update dist[v]

dist[v] = dist[u] + weight of edge uv

5.Then the input file is checked for negative cycle.

6.Two separate functions for printing the distance array and path traversal has been created.